Listify API Documentation

Введение

Это API предоставляет функционал для управления профилем пользователя, создания и взаимодействия с объявлениями, обмена сообщениями и администрирования. API поддерживает регистрацию, авторизацию и роли пользователей и администраторов.

Установка и настройка

1. **Установите зависимости**:

   ```
   npm install
   ```

2. **Настройте переменные окружения**: Создайте `.env` файл и укажите ключевые переменные, такие как:
   - `ADMIN_EMAIL`
   - `ADMIN_PASSWORD_HASH`
   - `JWT_SECRET`
   - `JWT_EXPIRES_IN`

3. **Запуск сервера**:

   ```
   npm start
   ```

Модель данных: User

Модель `User` имеет следующие поля:

- **id**: `INTEGER`, первичный ключ, автоинкрементируемый.
- **email**: `STRING`, уникальный, обязательно для заполнения, проверка на формат email.
- **password**: `STRING`, обязательно для заполнения, длина от 8 до 100 символов, хэшируется.
- **username**: `STRING`, обязательно для заполнения, длина от 3 до 20 символов.

Эндпоинты для управления профилем пользователя

Обновление профиля

URL: `/edit`

Метод: `PUT`

Аутентификация: Требуется

Описание: Обновляет профиль пользователя.

Тело запроса:

- `username`: `string`, новый никнейм пользователя (от 3 до 20 символов).
- `email`: `string`, новый email пользователя.

Ответы:

- `200 OK`: Профиль успешно обновлен.

```
{
"message": "Profile updated successfully",
"user": {
 "id": 1,
 "email": "newemail@example.com",
 "username": "newUsername"
}
}
```

- `404 Not Found`: Пользователь не найден.
- `500 Internal Server Error`: Ошибка на стороне сервера.

Смена пароля

URL: `/change-password`

Метод: `PUT`

Аутентификация: Требуется

Описание: Изменяет пароль пользователя после проверки текущего пароля.

Тело запроса:

- `currentPassword`: `string`, текущий пароль пользователя.
- `newPassword`: `string`, новый пароль (от 8 до 100 символов).

Ответы:

- `200 OK`: Пароль успешно изменен.

```
{
"message": "Password changed successfully"
```

```
}
```

- `400 Bad Request`: Текущий пароль неверен, новый пароль не соответствует требованиям.
- `404 Not Found`: Пользователь не найден.
- `500 Internal Server Error`: Ошибка на стороне сервера.

Эндпоинты для регистрации и авторизации пользователей

Регистрация

URL: `/register`

Метод: `POST`

Аутентификация: Не требуется

Описание: Регистрирует нового пользователя.

Тело запроса:

- `email`: `string`, email пользователя, должен быть уникальным.
- `password`: `string`, пароль (будет хэширован), минимальная длина — 8 символов.
- `username`: `string`, никнейм пользователя, длина от 3 до 20 символов.

Ответы:

- `201 Created`: Пользователь успешно зарегистрирован.

```
{
"message": "User registered successfully"
}
```

- `400 Bad Request`: Пользователь с указанным email уже существует.
- `500 Internal Server Error`: Ошибка на стороне сервера.

Авторизация

URL: `/login`

Метод: `POST`

Аутентификация: Не требуется

Описание: Авторизует пользователя, выдает JWT-токен для последующей аутентификации.

Тело запроса:

- `email`: `string`, email пользователя.
- `password`: `string`, пароль.

Ответы:

- `200 OK`: Успешная авторизация, возвращается токен и данные пользователя.

```
{
"message": "Login successful",
"token": "<JWT Token>",
"user": {
  "id": 1,
  "username": "username",
  "email": "user@example.com"
}
}
```

- `400 Bad Request`: Неверный email или пароль.
- `500 Internal Server Error`: Ошибка на стороне сервера.

Эндпоинт для авторизации администратора

Авторизация администратора

URL: `/login`

Метод: `POST`

Аутентификация: Не требуется

Описание: Авторизует администратора, выдавая JWT-токен для последующей аутентификации с ролью администратора.

Тело запроса:

- `email`: `string`, email администратора.
- `password`: `string`, пароль администратора.

Процесс:

- Проверка введенного `email` на соответствие значению переменной окружения `ADMIN_EMAIL`.
- Сравнение хэшированного `password` с сохраненным в `ADMIN_PASSWORD_HASH`.

Ответы:

- `200 OK`: Успешная авторизация, возвращается токен администратора.

```
{
"message": "Admin login successful",
"token": "<JWT Token>"
}
```

- `403 Forbidden`: Неверные учетные данные, доступ запрещен.
- `500 Internal Server Error`: Ошибка на стороне сервера.

Примечания:

- Токен содержит роль администратора (`role: 'admin'`), что может быть использовано для проверки доступа к защищенным ресурсам.
- Срок действия токена управляется переменной окружения `JWT_EXPIRES_IN` (по умолчанию — 1 час).

Основные маршруты API

Пользовательские маршруты

- **`/api/user`** – маршруты для управления данными пользователя (например, обновление профиля, изменение пароля).
- **`/api/auth`** – маршруты для регистрации и авторизации пользователей.

Административные маршруты

- **`/api/admin`** – маршруты для административного управления (например, модерация объявлений, управление пользователями).
- **`/api/admin-auth`** – маршруты для авторизации администратора.

Маршруты для работы с объявлениями

- **`/api/ads`** – маршруты для работы с объявлениями (создание, редактирование, удаление, поиск).

Маршруты для обмена сообщениями

- **`/api/messages`** – маршруты для отправки и получения сообщений между пользователями.

Пример структуры README:

1. **Введение** – краткое описание API и основных возможностей.
2. **Установка и настройка** – установка зависимостей, настройка переменных окружения.
3. **Модели данных** – описание ключевых моделей (`User` , `Ad` , `Message` и т.д.).
4. **Эндпоинты**:
    - **Пользовательские** (`/api/user` , `/api/auth`)

- **Административные** (`/api/admin` , `/api/admin-auth`)
- **Объявления** (`/api/ads`)
- **Сообщения** (`/api/messages`)

5. **Ответы на ошибки** – описание возможных кодов ошибок.
6. **Примеры запросов** – примеры JSON-запросов и ответов для основных операций.

Модель данных: Объявление (Ad) и Категория (Category)

Модель `Category`

- **id**: `INTEGER`, первичный ключ, автоинкрементируемый.
- **name**: `STRING`, обязательное, название категории.

Модель `Ad`

- **id**: `INTEGER`, первичный ключ, автоинкрементируемый.
- **userId**: `INTEGER`, внешний ключ, ссылается на пользователя (связь `Ad` с `User`).
- **title**: `STRING`, обязательное, заголовок объявления.
- **description**: `TEXT`, описание объявления.
- **categoryId**: `INTEGER`, внешний ключ, ссылается на категорию (связь `Ad` с `Category`).
- **price**: `DECIMAL`, обязательное, цена.
- **isSold**: `BOOLEAN`, статус продажи, по умолчанию `false`.

Эндпоинты для работы с объявлениями

Создание объявления

URL: `/create`

Метод: `POST`

Аутентификация: Требуется

Описание: Создает новое объявление.

Тело запроса:

- `title` : `string`, заголовок объявления.
- `description` : `string`, описание объявления.
- `categoryId` : `integer`, ID категории.
- `price` : `decimal`, цена объявления.

Ответы:

- `201 Created` : Объявление успешно создано, возвращает данные объявления.

- `500 Internal Server Error`: Ошибка на стороне сервера.

Получение объявлений пользователя

URL: `/my-ads`

Метод: `GET`

Аутентификация: Требуется

Описание: Возвращает все объявления, созданные текущим пользователем.

Поиск объявлений

URL: `/search`

Метод: `GET`

Описание: Позволяет искать объявления по различным параметрам.

Параметры запроса:

- `title`: `string`, текст для поиска в заголовках.
- `categoryId`: `integer`, ID категории.
- `minPrice`, `maxPrice`: `decimal`, диапазон цен.
- `sortBy`: `string`, критерий сортировки (`priceAsc`, `priceDesc`, `createdAt`).

Обновление объявления

URL: `/:id`

Метод: `PUT`

Аутентификация: Требуется

Описание: Обновляет объявление пользователя.

Удаление объявления

URL: `/:id`

Метод: `DELETE`

Аутентификация: Требуется

Описание: Удаляет объявление пользователя.

Отметка объявления как проданное

URL: `/:id/sold`

Метод: `PATCH`

Аутентификация: Требуется

Описание: Отмечает объявление как проданное.

Получение всех объявлений

URL: `/all`

Метод: `GET`

Описание: Возвращает все объявления.

Получение объявлений определенного пользователя

URL: `/user/:userId`

Метод: `GET`

Описание: Возвращает все объявления указанного пользователя.

Административные эндпоинты для удаления пользователей и объявлений

Удаление пользователя

URL: `/user/:userId`

Метод: `DELETE`

Аутентификация: Требуется (доступ только для администраторов)

Описание: Удаляет пользователя и все связанные с ним объявления.

Параметры пути:

- `userId`: `integer`, уникальный идентификатор пользователя.

Ответы:

- `200 OK`: Пользователь и связанные с ним объявления успешно удалены.

```
{
"message": "User deleted successfully, along with related ads"
}
```

- `404 Not Found`: Пользователь не найден.
- `500 Internal Server Error`: Ошибка на стороне сервера.

Удаление объявления

URL: `/ad/:adId`

Метод: `DELETE`

Аутентификация: Требуется (доступ только для администраторов)

Описание: Удаляет объявление по указанному ID.

Параметры пути:

- `adId`: `integer`, уникальный идентификатор объявления.

Ответы:

- `200 OK`: Объявление успешно удалено.

```
{
"message": "Ad deleted successfully"
}
```

- `404 Not Found`: Объявление не найдено.
- `500 Internal Server Error`: Ошибка на стороне сервера.

Модель данных: Сообщение (Message)

- **id**: `INTEGER`, первичный ключ, автоинкрементируемый.
- **senderId**: `INTEGER`, ID отправителя, внешний ключ, ссылается на `User`.
- **receiverId**: `INTEGER`, ID получателя, внешний ключ, ссылается на `User`.
- **adId**: `INTEGER`, ID объявления, к которому относится сообщение, внешний ключ, ссылается на `Ad`.
- **content**: `TEXT`, обязательное поле, содержание сообщения.

Эндпоинты для обмена сообщениями

Отправка сообщения

URL: `/send`

Метод: `POST`

Аутентификация: Требуется

Описание: Отправляет сообщение другому пользователю в рамках объявления.

Тело запроса:

- `adId`: `integer`, ID объявления.
- `content`: `string`, текст сообщения.

Ответы:

- `201 Created`: Сообщение успешно отправлено.

```
{
"message": "Message sent successfully",
"data": {
 "id": 1,
 "sender": { "id": 1, "username": "senderUser", "email": "sender@example.com"
},
```

```
 "receiver": { "id": 2, "username": "receiverUser", "email":
"receiver@example.com" },

 "content": "Message content",

 "createdAt": "2024-11-08T00:00:00.000Z"

}

}
```

- `404 Not Found`: Объявление не найдено.
- `400 Bad Request`: Попытка отправить сообщение самому себе.
- `500 Internal Server Error`: Ошибка на стороне сервера.

Получение сообщений по объявлению

URL: `/:adId`

Метод: `GET`

Аутентификация: Требуется

Описание: Получает все сообщения между пользователем и владельцем объявления для указанного объявления.

Параметры пути:

- `adId`: `integer`, ID объявления.

Ответы:

- `200 OK`: Успешно возвращает список сообщений.

```
[

{

 "id": 1,

 "sender": { "id": 1, "username": "senderUser", "email": "sender@example.com"
},

 "receiver": { "id": 2, "username": "receiverUser", "email":
"receiver@example.com" },

 "content": "Message content",

 "createdAt": "2024-11-08T00:00:00.000Z"

}
```

```
]
```

- `404 Not Found`: Объявление не найдено.
- `500 Internal Server Error`: Ошибка на стороне сервера.